Online Greedy Matching from a New Perspective

Lene M. Favrholdt[1]⋆ and Martin Vatshelle[2]

[1] Department of Mathematics and Computer Science, University of Southern Denmark, lenem@imada.sdu.dk

[2] Department of Informatics, University of Bergen, Martin.Vatshelle@ii.uib.no

Abstract. We introduce two new quality measures for online algorithms, the online worst-order and random-order ratios, that compare the online algorithms to optimal online algorithms instead of optimal offline algorithms. We apply these new measures as well as the competitive and random-order ratios to the matching problem, on general graphs and graphs of maximum degree 2.

1 Introduction

Consider a graph G with edge set E. A subset E' of the edges, such that no two edges in E' are adjacent, is called a matching in G. In the matching problem, the aim is to find a matching with as many edges as possible.

For the offline version of the problem, where the whole graph is known from the beginning, there are polynomial algorithms that solve the problem to optimality [7].

In the online version, the edges of the graph are revealed one by one. For each edge, the algorithm has to decide whether to include the edge in the matching, without knowledge of possible future edges.

The standard quality measure for online algorithms is the *competitive ratio* [11, 17, 20], which compares the online performance to the optimal offline performance, much like the approximation ratio for offline algorithms. It is easily seen that the natural greedy algorithm that always includes an edge in the matching, if it is not adjacent to an edge already included, has a competitive ratio of 0.5, and no deterministic algorithm has a better competitive ratio. This closes the competitive analysis of the problem. In this paper, we try to add more detail to the analysis of the online matching problem, using three other quality measures and looking at special graph classes.

The first alternative measure, the *random-order ratio*, was introduced in connection with the bin packing problem in [18]. It has also been used implicitly in papers on the offline matching problem [6]. For each input sequence, it considers the expected performance, assuming that all permutations of the sequence are equally likely.

The two other measures, the *online worst-order ratio* and the *online random-order ratio*, compare the online algorithm to an optimal online algorithm instead of an optimal offline algorithm. The idea behind is that, when we analyze algorithms for online problems, we should normalize their performance by the hardness of the online problem, not the hardness of the offline problem.

The two online measures are inspired by [14], which introduces the *online chromatic number* of a graph. The online chromatic number of a graph G is the number of colors it takes the best possible online algorithm (with respect to G) to color G. It can be assumed that the optimal online algorithm knows the graph beforehand, but not the order in which the vertices arrive. For the coloring problem, the online worst-order ratio of an online algorithm A is the worst-case ratio of the number of colors used by A to the online chromatic number. The online random-order ratio is the same as the online worst-order ratio except that, for each input sequence, the average (instead of worst-case) performance over all permutations of the sequence is considered. The online worst-order and random-order ratios

⋆ Supported in part by the Danish Agency for Science, Technology and Innovation (FNU).

can also be seen as relaxations of the worst-order ratio [2] and random-order ratio. The online worst-order and random-order ratios are formally defined in Section 2.

1.1 Results

We introduce two new quality measures for online algorithms, the online worst-order and random-order ratios.

We analyze the competitive ratio (CR), random-order ratio (RR), online worst-order ratio (OWR), and online random-order ratio (ORR). The results are summarized in Tables 1–3. The left columns give results on the greedy algorithm, and the right columns show bounds on the best possible deterministic online performance. We first look at general graphs (Table 1). Then we move on to the special case of graphs of maximum degree 2 (Table 2). We also consider the even more special case, where the graph consists of one long path or cycle (Table 3). For this case, our results for the online measures are the same as for graphs of maximum degree 2.

We implemented an algorithm which is always at least as good as any online algorithm. We compare this algorithm to Greedy on paths of lengths up to 100,000. Results obtained with this simulation are marked with an $*$ in the tables.

Greedy	Opt(Det)
$\mathrm{CR} = 0.5$	$\mathrm{CR} = 0.5$
$\mathrm{RR} = 0.5$	$0.5 \leq \mathrm{RR} \leq 0.78\overline{3}$
$\mathrm{OWR} = 0.5$	$0.5 \leq \mathrm{OWR} \leq 0.\overline{6}$
$\mathrm{ORR} = 0.5$	$0.5 \leq \mathrm{ORR} < 1$

Table 1. General graphs

Greedy	Opt(Det)
$\mathrm{CR} = 0.5$	$\mathrm{CR} = 0.5$
$0.8\overline{2} \overset{*}{\leq} \mathrm{RR} \leq 0.8\overline{2}$	$0.8\overline{2} \overset{*}{\leq} \mathrm{RR} \leq 0.8\overline{2}$
$\mathrm{OWR} = 1$	$\mathrm{OWR} = 1$
$0.993 \overset{*}{\lesssim} \mathrm{ORR} \leq 1$	$0.993 \overset{*}{\lesssim} \mathrm{ORR} \leq 1$

Table 2. Graphs of maximum degree 2

Greedy	Opt(Det)
$\mathrm{CR} = 0.\overline{6}$	$\mathrm{CR} = 0.\overline{6}$
$\mathrm{RR} \approx 0.865$	$0.865 \lesssim \mathrm{RR} \overset{*}{\leq} 0.87$
$\mathrm{OWR} = 1$	$\mathrm{OWR} = 1$
$0.993 \overset{*}{\lesssim} \mathrm{ORR} \leq 1$	$0.993 \overset{*}{\lesssim} \mathrm{ORR} \leq 1$

Table 3. Long paths or cycles

For graphs of maximum degree 2, Greedy is optimal with respect to the online worst-order ratio. This adds some detail to the information we get from the competitive ratio. The fact that Greedy has an optimal competitive ratio says that, for any deterministic online algorithm A, there is a graph where A does as bad as Greedy does on its worst graph.

In a sense, the result $\mathrm{OWR}(\mathrm{Greedy}) = 1$ on graphs of maximum degree 2 says that, on *any* graph of maximum degree 2, Greedy does at least as well as any deterministic online algorithm.

It seems that, on graphs of maximum degree 2, Greedy is also optimal with respect to the online random-order ratio, but we have not been able to prove it. We prove that, on a path of length 9, P_9, there is an online algorithm with a better random-order performance than Greedy. However, this does not disprove that the online random-order ratio of Greedy could be 1, since the random-order ratio is an asymptotic measure, and the proof does not generalize to arbitrarily long paths or more copies of P_9.

1.2 Related Work

In [18] it was shown that the random-order ratio of the Best-Fit bin packing algorithm lies between 1.08 and 1.5. Thus, the random-order ratio is lower than the competitive ratio of 1.7. In contrast, it was recently shown in [15] that the random-order ratio of the Next-Fit bin packing algorithm is 2, which is the same as the competitive ratio.

In [6] a randomized version of the greedy matching algorithm is analyzed. The algorithm considers the edges in a random order. On general graphs, this strategy leads to a competitive ratio of $\frac{1}{2}$, just like the deterministic greedy algorithm, but on some graph classes, the ratio is significantly better: On forests, the ratio is approximately 0.769, and on planar graphs, it lies between $\frac{6}{11}$ and $\frac{11}{15}$. Note that these ratios correspond to the random-order ratio of the greedy algorithm.

In [5, 12–14, 19], online algorithms are sought that use a number of colors which is a function depending only on the online chromatic number, for any graph in a given class. Such algorithms are called online competitive for that graph class. It is not known whether online competitive algorithms exist for all graph classes; it is not even settled for the class of bipartite graphs.

Several papers have analyzed the (relative) worst-order ratio for various online problems [1–4, 8, 9].

An overview of online bipartite matching, analyzed with the competitive ratio, is given in [16].

2 Quality Measures

In this section, we formally define the four quality measures. First, we introduce some notation.

Let OPT denote an optimal offline matching algorithm. For any algorithm A and any sequence E of edges, we let $A(E)$ denote the the number of edges in the matching constructed by A when given E as input. For any graph $G = (V, E)$, let $A_{\mathrm{W}}(G)$ denote the worst-case performance of A on G, over all permutations of E. More precisely,

$$A_{\mathrm{W}}(G) = \min_{\sigma \in \mathcal{S}} \{A(\sigma(E))\}\,,$$

where $\mathcal{S}$ denotes the set of permutations on $|E|$ elements. We are now ready to define the competitive ratio:

Definition 1 (Competitive Ratio). *For any online matching algorithm A, the* competitive ratio *of A is*

$$\mathrm{CR}(A) = \sup\{c \mid \exists b\colon \forall G\colon\ A_{\mathrm{W}}(G) \geq c\,\mathrm{OPT}(G) - b\}\ .$$

For the definition of the random-order ratio we use the following further notation: Let $A_{\mathrm{E}}(G)$ denote the average number of edges in the matching produced by A, over all permutations of the edges of G, i.e.,

$$A_{\mathrm{E}}(G) = \frac{1}{|\mathcal{S}|} \sum_{\sigma \in \mathcal{S}} A(\sigma(E))\,.$$

Definition 2 (Random-Order Ratio). *For any online matching algorithm A, the* random-order ratio *of A is*

$$\mathrm{RR}(A) = \sup\{c \mid \exists b\colon \forall G\colon A_{\mathrm{E}}(G) \geq c\,\mathrm{OPT}(G) - b\}\ .$$

For the definition of the two online measures, we let $\mathcal{O}$ denote the set of all online matching algorithms.

Definition 3 (Online Worst-Order Ratio). *For any online matching algorithm A, the* online worst-order ratio *of A is*

$$\mathrm{OWR}(A) = \sup\{c \mid \exists b\colon \forall G\colon \forall O \in \mathcal{O}\colon A_{\mathrm{W}}(G) \geq c\,O_{\mathrm{W}}(G) - b\}\ .$$

One of the differences between the competitive ratio and the online worst-order ratio is the following. With the competitive ratio, the online algorithm is competing against an optimal offline algorithm. With the online worst-order ratio, for each graph, the online algorithm is competing against a best possible online algorithm for that graph. (This is exactly the difference between the worst-order ratio and the online worst-order ratio.)

Note that considering a separate online algorithm A_G for each graph G does not necessarily mean that A_G can perform as well as an offline algorithm, since it is not given a mapping between the arriving edges and the edges of the final graph.

The analogy between the random-order ratio and the online random-order ratio is the same as between the worst-order ratio and the online worst-order ratio:

Definition 4 (Online Random-Order Ratio). *For any online matching algorithm A, the* online random-order ratio *of A is*

$$\mathrm{ORR}(A) = \sup\{c \mid \exists b\colon \forall G\colon \forall O \in \mathcal{O}\colon A_{\mathrm{E}}(G) \geq c\,O_{\mathrm{E}}(G) - b\}\ .$$

3 General Graphs

It is well-known that the greedy algorithm has an optimal competitive ratio of 0.5:

Theorem 1. *The competitive ratio of Greedy is* $\mathrm{CR}(\mathrm{Greedy}) = 0.5$*, and this is optimal among deterministic online algorithms.*

Proof. For a graph G, let M be the optimal matching. Every edge of M that is not picked by Greedy is adjacent to one edge picked by greedy. Since an edge can be adjacent to at most two edges of M, Greedy gets at least $\frac{|M|}{2}$ edges.

Let G be a collection of k copies of P_3, i.e., k independent paths of length three. Then $|M| = 2k$. If all middle edges of the P_3's arrive first, Greedy gets k edges, and has to reject all the $2k$ edges in M. □

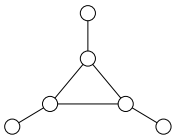
In [6] it is proven that, even if Greedy considers the edges in a random order, there is a graph, where the expected number of edges accepted by Greedy is only half the number of edges accepted by an optimal offline algorithm. This immediately gives Theorem 2 below. The graph considered in the proof is a clique with one additional edge sticking out from each clique vertex.

Theorem 2. *The random-order ratio of* Greedy *is* $\mathrm{RR}(\mathrm{Greedy}) = 0.5$.

Theorem 3. *Any online algorithm A has a random-order ratio of*

$$\mathrm{RR}(A) \leq 47/60 = 0.78\overline{3}.$$

Proof. Consider the graph consisting of a triangle with an extra edge sticking out from each vertex:



The triangle edges are called inner edges, the other three are called outer edges. Clearly, an optimal matching consists of the three outer edges.

We show that, on this graph, an optimal online algorithm accepts the first edge. Consider any online algorithm A.

Case 1: A does not accept the first edge.

If the first edge is an *outer* edge, A will end up with a matching of at most two edges. This happens with probability $\frac{1}{2}$.

Now consider the case that the first edge is an *inner* edge. When the second edge arrives, the algorithm can distinguish only two cases: the edge is adjacent or nonadjacent to the first edge.

With probability $\frac{4}{5}$, the second edge will be adjacent to the first edge, and when this happens, the probability that the second edge is an inner edge is $\frac{1}{2}$. If the edge is an inner edge, and A accepts it, A will get at most two edges. If the edge is an outer edge, and A does not accept it, A will get at most two edges. Thus, if the second edge is a neighbor of the first edge, A gets at most two edges with probability at least $\frac{1}{2}$.

Hence, if A does not accept the first edge, the total probability of getting at most two edges is at least $\frac{1}{2} + \frac{1}{2} \cdot \frac{4}{5} \cdot \frac{1}{2} = \frac{7}{10}$. Thus, the expected number of edges in A's matching is at most $\frac{7}{10} \cdot 2 + \frac{3}{10} \cdot 3 = \frac{23}{10}$.

Case 2: A accepts the first edge.

If the first edge is an *inner* edge, A ends up with a matching of at most two edges. This happens with probability $\frac{1}{2}$.

Now consider the case that the first edge is an *outer* edge. We consider the following two subcases:

Case 2a: The second edge is adjacent to the first edge. This happens with probability $\frac{2}{5}$.

With probability $\frac{1}{2}$ the third edge is adjacent to the second edge, but not to the first edge. In this case, it is equally likely to be an inner or outer edge. Hence, with probability $\frac{1}{2}$ A gets at most two edges.

Case 2b: The second edge is not adjacent to the first edge. This happens with probability $\frac{3}{5}$.

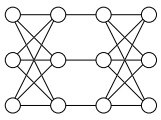
With probability $\frac{2}{3}$, the second edge is an outer edge. Thus, the best strategy is to accept the edge. But this still leaves a probability of $\frac{1}{3}$ of getting at most 2 edges.

Hence, if A accepts the first edge, the total probability of getting at most 2 edges is at least $\frac{1}{2} + \frac{1}{2}(\frac{2}{5} \cdot \frac{1}{2} \cdot \frac{1}{2} + \frac{3}{5} \cdot \frac{1}{3}) = \frac{13}{20}$. This gives an expected number of accepted edges of at most $\frac{13}{20} \cdot 2 + \frac{7}{20} \cdot 3 = \frac{47}{20}$.

Thus, the best strategy is to accept the first edge, in which case the expected number of accepted edges is at most $\frac{47}{20}$. Since the optimal number of edges is three, this gives a ratio of at most $\frac{47}{60}$. □

Theorem 4. *The online worst-order ratio of* Greedy *is* $\mathrm{OWR}(\mathrm{Greedy}) = 0.5$.

Proof. Consider the graph G_k that contains two copies of $B_{k,k}$, $B_1 = (L_1 \cup R_1, E_1)$ and $B_2 = (L_2 \cup R_2, E_2)$. Vertex i in R_1 has an edge to vertex i in L_2, $i = 1, 2, \ldots, k$. The graph G_3 looks like this:



If the edges connecting B_1 and B_2 are given first, Greedy will accept these k edges and no other edges.

Consider the algorithm A that accepts an edge if and only if there is already a path of length three between its endpoints. This algorithm will not accept any of the edges connecting B_1 and B_2. Hence, we can prove that A accepts at least $k-1$ edges from each B_i, $i = 1, 2$. To this end, we define two subgraphs of B_i. The *green* graph is the subgraph of B_i induced by the endpoints of the accepted edges from E_i. The *red* graph is the subgraph of B_i induced by the remaining vertices of $L_i \cup R_i$.

L_i and R_i can have at most one red vertex each. Otherwise, there would be a four-cycle in the red graph. This would be a contradiction, since when one of these four edges arrived, its endpoints would already be connected by a path of length three, and thus it would be accepted. It follows that L_i and R_i have at most one red vertex each, and hence the green graph has at least $k-1$ edges. □

Theorem 5. *Any online algorithm A has an online worst-order ratio of*

$$\mathrm{OWR}(A) \leq \frac{2}{3}$$

Proof. Consider the graph G_k described in the proof of Theorem 4 and the graph M_k consisting of k independent edges. No online algorithm can be optimal on both these graphs.

Consider any algorithm A. Assume that the algorithm is given k independent edges. If A accepts only $\frac{2}{3}$ of the edges, then if the sequence stops after these k edges, A gets only $\frac{2}{3}$ as many edges as the algorithm that accepts all of them. On the other hand, if A accepts more than $\frac{2}{3}k$ edges, and if the sequence continues until all of G_k has been given, A gets at most $\frac{4}{3}k$ edges. According to the proof of Theorem 4, there is an online algorithm that accepts at least $2k-2$ edges of G_k, on any permutation of the edges. This gives a ratio that tends to $\frac{2}{3}$ as k tends to infinity.

Note that the proof goes through also for randomized algorithms, with "number of edges" replaced by "expected number of edges". Hence, the result holds for randomized as well as deterministic algorithms. □

Theorem 6. *The online random-order ratio of* Greedy *is* $\mathrm{ORR}(\mathrm{Greedy}) = 0.5$.

Proof. The lower bound follows from Theorem 2. For the upper bound, consider the graph G_n defined in the following way. The graph contains an n-clique Q_n. Each vertex in Q_n is connected to exactly one vertex outside the clique, and no two vertices in Q_n are connected to the same vertex outside the clique. Thus, the graph has exactly $2n$ vertices and $m = \frac{1}{2}(n^2+n)$ edges. The n clique vertices are also called *inner vertices*, and the remaining n vertices are called *outer vertices*. Similarly, the $\frac{1}{2}(n^2-n)$ clique edges are called *inner edges*, and the n edges connecting inner vertices to outer vertices are called *outer edges*.

The optimal thing to do is, of course, to reject the inner edges and accept all the outer edges. This will give a matching of exactly n edges. However, for each outer edge, about $\frac{1}{2}n$ inner edges will arrive, so Greedy will accept many inner edges. By Theorem 2, Greedy gets only $\frac{1}{2}n + o(n)$ edges in total.

We will describe an algorithm that rejects all arriving edges, until it can make educated guesses as to which edges are inner edges and which are outer edges. The idea is to try to reject the first $\Theta(n \log n)$ edges, and after that accept exactly those edges that are adjacent to at most one vertex that has already been seen. We prove that this strategy is expected to loose only $O(\log n)$ outer edges among the $\Theta(n \log n)$ edges that are rejected unconditionally and $o(n)$ outer edges among the remaining edges. Finally, we show how to decide when $\Theta(n \log n)$ edges have arrived, without knowing n from the beginning. This will prove the upper bound.

In a random permutation, each edge has a probability of $n \log n/m$ of being among the first $\Theta(n \log n)$ edges. Hence the expected number of outer edges among the first $\Theta(n \log n)$ edges is $\Theta(\log n)$.

When r edges have arrived, let $x(r)$ be the expected number of inner vertices which are not among the endpoints of the r first edges of input. We call these vertices *unseen* inner vertices. There are two ways an outer edge can be rejected, either it was among the $\Theta(n \log n)$ first edges or it was connected to one of at most $x(n \log n)$ unseen inner vertices. We prove that $x(n \log n) \in o(n)$.

Assume for the purpose of contradiction that $x(n \log n)$ is $\Omega(n)$. Then for sufficiently large n, there exists a constant ε such that $x(n \log n) \geq \varepsilon n$. Therefore, for each inner edge among the first $n \log n$ edges, the probability of being incident to some inner vertex not previously seen is at least ε. Moreover, the number of inner edges among the $n \log n$ first edges is $\Omega(n \log n)$. Thus, for sufficiently large n, there exists a constant ε' such that $n - x(n \log n) \geq \varepsilon \cdot \varepsilon' n \log n$, but for sufficiently large n, this is a contradiction. Hence, we conclude that $x(n \log n)$ is $o(n)$.

The only thing left to prove is how to reject the $\Theta(n \log n)$ first edges without knowing n. Let $H_r = (V_r, E_r)$ be the subgraph of G_n, where E_r contains the r first edges of the input, and V_r contains the vertices adjacent to at least one of the r first edges. Then the vertices of H_r will be the $n - x(r)$ seen inner vertices plus the $O(\log n)$ seen outer vertices. We want to show that when the average degree of H_r is $\Theta(\log |V_r|)$ then the expected value of $|E_r|$ is $\Theta(n \log n)$.

Clearly, when the average degree of H_r is $\Theta(\log |V_r|)$, the number of seen edges is $O(n \log n)$. Hence it suffices to show: If the average degree of H_r is $\Omega(\log |V_r|)$ then the expected value of $|E_r|$ is $\Omega(n \log n)$. Or the contrapositive: If $r = o(n \log n)$, then the expected average degree of H_r is $o(\log |V_r|)$.

Let $r = o(n \log n)$, then the expected number of outer edges is $o(\log n)$. We divide into two cases, $r \leq \frac{n}{4}$ and $r > \frac{n}{4}$. When $r \leq \frac{n}{4}$ the probability of the next edge to see an unseen node is at least $\frac{1}{2}$. Hence, the expected number of seen vertices is at least $\frac{1}{2}r$, and the expected average degree is at most 4. If $r > \frac{n}{4}$, the expected number of seen vertices is at least $\frac{n}{8}$, and hence the expected average degree is $o(\log n)$. Hence, the algorithm can reject until the average degree is $\log |V_r|$; then the expected number of rejected edges will be $\Theta(n \log n)$. □

Thus, in a sense, Greedy does not look better competing against online algorithms than offline algorithms. This is not very surprising. It is to be expected that, as for graph coloring, one has to look at more restricted graph classes to see the difference between the two measures.

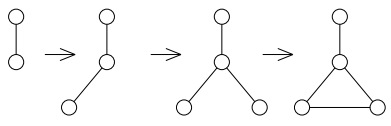
In Theorem 5 we used two graphs and showed that any online algorithm would fail on at least one of them. The important part was that the two input sequences were indistinguishable. For random-order measures we have to prove that two graphs, or classes of graphs, are indistinguishable for all online algorithms.

In this proof the graphs is a collection of stars with different size. For each component we add a constant number of extra edges in such a way that no online algorithm can get these edges with out knowing where they are placed. Therefore almost all sequences look equal in the start.

Theorem 7. *No online algorithm A has an online random-order ratio of 1.*

Proof. We consider two graphs, the graph from the proof of Theorem 3 (T_3) and a triangle with just one edge sticking out (T_1). We prove that no online algorithm can be online random-order optimal on both graphs.

As in the proof of Theorem 3, we divide the edges into inner and outer edges. Thus, T_1 has one outer edge, and T_3 has three outer edges. Consider the sequence I starting with an outer edge e, followed by the two neighbors of e, and concluded by the remaining inner edge:



This sequence defines all of T_1 or a subgraph of T_3. Clearly, if the final graph is T_1, the first and the last edge of I should be accepted. However, if the final graph is T_3, the last edge of I should be rejected.

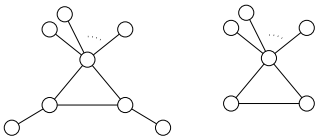
By case analysis, it is easily proven that Greedy is online random-order optimal for T_1, with $\frac{3}{2}$ accepted edges on average. Similarly, it is not difficult to prove that, for T_3, the optimal strategy is Greedy, except that if the input sequence starts with sequence I, the fourth edge should be rejected. This gives $\frac{47}{20}$ accepted edges on average.

For the *absolute* ratio, this gives a ratio of at most $\frac{140}{141}$:

If the final graph is T_3, the probability that the sequence I will be a prefix of the input sequence is $\frac{1}{2} \cdot \frac{2}{5} \cdot \frac{1}{4} \cdot \frac{1}{3} = \frac{1}{60}$. Thus, if the last edge of I is accepted, the expected number of accepted edges will be at most $\frac{47}{20} - \frac{1}{60} = \frac{140}{60}$. For the optimal online algorithm for T_3, the expected number of edges is $\frac{47}{20} = \frac{141}{60}$. This gives a ratio of $\frac{140}{141}$.

If the final graph is T_1, the probability of the sequence I is $\frac{1}{4} \cdot \frac{2}{3} \cdot \frac{1}{2} = \frac{1}{12}$. Thus, if the last edge is rejected, the expected number of edges is at most $\frac{3}{2} - \frac{1}{12} = \frac{17}{12}$. For the optimal online algorithm for T_1, the expected number of edges is $\frac{3}{2} = \frac{18}{12}$. This gives a ratio of $\frac{17}{18}$, which is smaller than $\frac{140}{141}$.

For the *asymptotic* ratio, we construct an infinite family of graphs. Each graph consists of copies of T_1 and T_3 with a lot of extra outer edges attached to a vertex that already has an outer edge:



In a graph with n copies of T_1 and T_3 in total, the numbers of extra outer edges could be n^2, n^3, …, n^n. With very high probability a large fraction of these extra edges have arrived before the online algorithm has to make decisions about the third inner edge. Thus, with high probability, the optimal online algorithm will be able to sort them by the number of extra outer edges, and thus infer which components will end up as a T_1 and which will end up as a T_3. □

4 Graphs of Maximum Degree 2

In this section, we consider graphs that consist only of paths and cycles. We let P_i, $i \in \mathbb{N}$, denote a path of length i.

Theorem 8. *On graphs of maximum degree 2, the competitive ratio of* Greedy *is*

$$\mathrm{CR}(\mathrm{Greedy}) = \frac{1}{2}\,.$$

Proof. The result follows from the proof Theorem 1. □

We will now investigate the performance of an optimal algorithm

Lemma 1. *For a path P_i and any deterministic online algorithm A, there exists an order of the input where A gets at most $\lceil \frac{i}{3} \rceil$ edges.*

Proof. For paths of length at most 4 this is easy to check. Assume the lemma holds for all lengths less than k. Consider any deterministic online algorithm A. Assume that a path P_{i-3} of length $i-3 < k$ has been given and that A has accepted at most $\lceil \frac{i-3}{3} \rceil$ edges. We show how to extend the path by three additional edges such that A accepts at most one of these edges. The first of the three edges is not connected to P_{i-3}, i.e., it is the last or second last edge of P_i. A can either accept or reject the edge.

1. If accept, then the edge is the second last edge of P_i. The last and third last edges will be given last, and none of these can be included in the matching.

2. If reject, then the edge is the last edge of P_i. The second last and third last will be given next, and from these A gets at most one.

□

Lemma 2. *For a path P_i,* Greedy *always gets at least $\lceil \frac{i}{3} \rceil$ edges.*

Proof. Assume not, then 3 consecutive internal edges must have been rejected by Greedy, or two edges at the end of the path. But Greedy only rejects an edge if at least one neighboring edge has been accepted. □

Theorem 9. *For graphs of maximum degree 2,* $\mathrm{OWR}(\mathrm{Greedy}) = 1$.

Proof. The result follows directly from Lemmas 1 and 2. □

If we want to prove that $\mathrm{ORR}(\mathrm{Greedy}) = 1$, we should not try to prove that Greedy always does at least as well as any online algorithm:

Theorem 10. *The absolute online random-order ratio of* Greedy *is strictly less than 1.*

Proof. There is an algorithm that performs slightly better than Greedy on P_9:

Let the edges $e_1, e_2, \ldots, e_9$ be the edges of P_9, such that e_1 and e_9 are the endedges, e_2 and e_8 are their neighbors, and so on. Assume that the first six edges to arrive form two vertex disjoint paths of length three, and that the middle edge in each path has been included in the matching. Furthermore, assume that the seventh edge to arrive is not adjacent to any of these paths. Then, with probability $\frac{1}{3}$ the new edge is the middle edge, e_5, and with probability $\frac{2}{3}$ it is one of the endedges. If it is the first (or last) edge of the path, it is safe to reject it, since the second (or second last) edge can be accepted instead. If it is the middle edge, it pays to reject it, because then its two neighbors can be accepted.

The algorithm A that behaves like Greedy except for the case described above has a better expected performance than Greedy. For the sequence to start as described, with the seventh edge being the middle edge of the path, the first six edges to arrive must be exactly the first and last three edges of the path, e_2 must arrive before e_1 and e_3, e_8 must arrive before e_7 and e_9, and e_5 must arrive before e_4 and e_6. Hence, the probability of the sequence starting as described with the seventh edge being the middle edge is

$$\frac{1}{\binom{9}{6}} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{2268}.$$

Thus, the expected number of edges accepted by A is $\frac{1}{2268}$ larger than the expected number of edges accepted by Greedy. □

The above theorem indicates that it could be challenging to show $\mathrm{ORR}(\mathrm{Greedy}) = 1$. We found no technique that works in general. However, for a fixed path length, there is a simple way of calculating the expected size of the matching found by Greedy, using dynamic programming.

We can use a similar way of dynamic programming to give an upper bound on the performance of an optimal online algorithm. We will now briefly describe the algorithm before we start talking about the results.

An online algorithm can not tell which edge it is given, but if one or both neighboring edges are previously seen an online algorithm could keep track of this. There are therefore 3 possible states for each endpoint of a yet unseen edge: The neighboring edge has not been seen, it has been seen and is in the matching or it has been seen and was rejected. We call those edges arriving before both neighboring edges *independent* edges. If an online algorithm gets an edge which is not independent the greedy strategy is optimal. The strategy of an optimal online algorithm is therefore only uncertain when facing an independent edge.

The graph induced by the set of unseen edges will be a collection of paths. We call them *upaths*. For each upath there are 3 possible states for each endpoint, but because of

symmetry we only need 6 states in total for each upath. If the online algorithm knew the length of the upath containing the given edge and the state of its endpoints it would be able to make a choice better than any online algorithm could do.

We keep a table with one entry for each pair of length and state. Since the edges come in a random order, each edge of a upath has equal probability of being the next one. For non-independent edges we know Greedy is optimal, and for independent edges there are only two possible strategies: reject or accept. In either case, given the length of the upath and the states of its endpoints, it is just to sum, over all edges in the upath, the expected number of matched edges in the two resulting shorter upaths times the probability for the edge, and use the strategy giving the best expected value. This semi-online algorithm is called onlOPT, since it gives an upper bound on the best possible online performance.

If we want the expected number of edges for Greedy we do the same, but do not make a choice of the best strategy.

As for the results of this simulation there are several interesting points: How well does Greedy do compared to the optimal solution? We calculated the random-order ratio for each path length up to $100,000$. The worst random-order ratio for both Greedy and onlOPT was for paths of length 5, where they had the same ratio of $\frac{37}{45} = 0.8\overline{2}$. This indicates:

Simulation Result 1 $\mathrm{RR}(\mathrm{Greedy}) = \mathrm{RR}(\mathrm{onlOPT}) = 0.8\overline{2}$.

Lemma 3. *For graphs of max degree* 2, $\mathrm{RR}(\mathrm{Greedy}) \leq 0.8\overline{2}$.

Proof. Let the input be a collection of P_5's. Let the edges $e_1, e_2, \ldots, e_5$ be the edges of a P_5, such that e_1 and e_5 are the endedges, e_2 and e_4 are their neighbors, and e_3 is the middle edge. Greedy will always get at least 2 edges and at most 3, so we only need to calculate the probability of getting 2 edges. If the order of edges starts with (e_2),(e_4), (e_1,e_4), (e_5,e_2), (e_1,e_2,e_4), or (e_5,e_4,e_2), Greedy gets 2 edges, and else Greedy gets 3 edges. The probability of getting 2 edges is: $2 \cdot \frac{1}{5} + 2 \cdot \frac{1}{5} \cdot \frac{1}{4} + 2 \cdot \frac{1}{5} \cdot \frac{1}{4} \cdot \frac{1}{3} = \frac{2}{5} + \frac{1}{10} + \frac{1}{30} = \frac{8}{15}$. Hence, the expected number of edges is: $\frac{8}{15} \cdot 2 + \frac{7}{15} \cdot 3 = \frac{37}{15}$. Since OPT gets 3 edges, $\mathrm{RR}(\mathrm{Greedy}) \leq \frac{37}{45} = 0.8\overline{2}$ □

For long paths, we get that $\mathrm{RR}(\mathrm{Greedy})$ tends to $0.864\ldots$ and $\mathrm{RR}(\mathrm{onlOPT})$ tends to $0.869\ldots$, and we find the following relationship between the performance of the two algorithms:

Simulation Result 2 $\mathrm{Greedy}(n) \approx 0.9948\ldots \cdot \mathrm{onlOPT}(n) + 0.0168$.

Recall that for Greedy we calculated the exact expected value, but the value for onlOPT is only an upper bound on the best possible online performance. Hence, Greedy could be optimal for long paths, but we do not know how to prove that.

5 Long Paths or Cycles

For long paths and cycles, many of the results from the previous section hold, and some can be improved.

Theorem 11. *On long paths and cycles, any deterministic online algorithm A has a competitive ratio of* $\mathrm{CR}(A) \leq \frac{2}{3}$.

Proof. This follows from Lemma 1. □

Theorem 12. *On long paths and cycles, the competitive ratio of* Greedy *is*

$$\mathrm{CR}(\mathrm{Greedy}) = \frac{2}{3}\,.$$

Proof. This follows from Lemma 2 and Theorem 11. □

The following result follows directly from Theorems 11 and 12.

Corollary 1. *On long paths and cycles,* $\mathrm{OWR}(\mathrm{Greedy}) = 1$.

For paths and cycles, online matching is equivalent to unfriendly seating arrangements. One can view edges as seats. The unfriendly seating problem was studied by Dave Freedman and Larry Shepp in 1964 [10].

Theorem 13. *If the edges of a long path or cycle are given in a random order,* Greedy *will on average pick approximately 43% of the edges.*

Since the optimal solution matches half the edges, this immediately gives the following result.

Corollary 2. *On long paths and cycles, the random-order ratio of* Greedy *is* $\mathrm{RR}(\mathrm{Greedy}) \approx 0.86$.

We see that this is consistent with the results from our simulation. The simulation also showed that no deterministic online algorithm has random-order ratio ≥ 0.87. We therefore conclude the following:

Simulation Result 3 *On long paths and cycles,* Greedy *has an online random-order ratio of* $\mathrm{ORR}(A) \geq 0.994$.

Interestingly enough there is another algorithm that performs just as well. The algorithm keeps track of what Greedy would have done, and does the following. If Greedy would have taken an edge, the edge is rejected, and otherwise it is accepted if possible. We call this algorithm revGreedy.

Lemma 4. *If the edges of a long cycle or path are given in a random order,* revGreedy *will on average pick approximately* 43% *of the edges.*

Proof. revGreedy picks among the edges that Greedy left out, i.e., a collection of paths of length 1 or 2. For each of these paths, revGreedy picks one edge. Therefore, Greedy picks at most one more edge than revGreedy. The result now follows from Theorem 13. □

Note that there is an equivalent definition of revGreedy: For each edge, if some neighbor edge has been seen before, accept if possible. Else reject.

6 Conclusion and Future Work

We introduced two new quality measures, where we use the best possible online performance as the reference point instead of the best possible offline performance. The analysis gave some additional evidence that Greedy is the best choice for online matching. And none of the measures gave any evidence indicating that Greedy is not optimal.

However, we found some specific instances of short paths where Greedy is not the best possible online algorithm. Moreover, the new online random-order measure opens for the possibility that Greedy is not optimal. Hence, it is an important question whether $\mathrm{ORR}(\mathrm{Greedy}) = 1$ for graphs of maximum degree 2.

Furthermore, there is another algorithm, very different from Greedy, with the same performance as Greedy on long paths and cycles. This was a surprising result, but the new algorithm is clearly not as good as Greedy on general graphs.

We are currently working on generalizing and applying the new measures to online weighted matching.

We would also like to continue a discussion on whether the new measures capture the optimality of an online algorithm or not. The reason we needed a new measure was that competing against the optimal offline algorithm was too tough for an online algorithm. For a given graph, it is most likely a highly specialized algorithm that is the optimal online algorithm, and thus does arbitrarily bad on other inputs. Would it be possible to only compete against a subset of the online algorithms that satisfy some requirement of generality? And still be able to calculate the value of such a measure?

References

1. Joan Boyar, Martin R. Ehmsen, and Kim S. Larsen. Theoretical evidence for the superiority of LRU-2 over LRU for the paging problem. In *Approximation and Online Algorithms*, pages 95–107, 2006.
2. Joan Boyar and Lene M. Favrholdt. The relative worst order ratio for on-line algorithms. *ACM Transactions on Algorithms*, 3(22), 2007.
3. Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. The relative worst-order ratio applied to paging. *Journal of Computer and System Sciences*, 73:818–843, 2007.
4. Joan Boyar and Paul Medvedev. The relative worst order ratio applied to seat reservation. In *Ninth Scandinavian Workshop on Algorithm Theory*, pages 90–101, 2004.
5. Hajo J. Broersma, Agostino Capponi, and Daniël Paulusma. A new algorithm for on-line coloring bipartite graphs. *SIAM Journal on Discrete Mathematics*, 22(1):72–91, 2008.
6. Martin Dyer and Alan Frieze. Randomized greedy matching. *Random Structures and Algorithms*, 2:29–26, 1991.
7. Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
8. Martin R. Ehmsen, Lene M. Favrholdt, Jens S. Kohrt, and Rodica Mihai. Comparing First-Fit and Next-Fit for online edge coloring. In *19th International Symposium on Algorithms and Computation*, pages 89–99, 2008.
9. Leah Epstein, Lene M. Favrholdt, and Jens S. Kohrt. Separating online scheduling algorithms with the relative worst order ratio. *Journal of Combinatorial Optimization*, 12(4):362–385, 2006.
10. Dave Freedman and Larry Shepp. An unfriendly seating arrangement. *SIAM Review*, 6(2):180–182, 1964.
11. Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal*, 45:1563–1581, 1966.
12. András Gyárfás, Zoltán Király, and Jenö Lehel. Online 3-chromatic graphs II. Critical graphs. *Discrete Mathematics*, 177:99–122, 1997.
13. András Gyárfás, Zoltán Király, and Jenö Lehel. Online competitive coloring algorithms. Technical Report TR-9703-1, Eötvös Loránd University, Institute of Mathematics, 1997. available online at http://www.cs.elte.hu/tr97/tr9703-1.ps.
14. András Gyárfás and Jenö Lehel. First-fit and on-line chromatic number of families of graphs. *Ars Combinatoria*, 29C:168–176, 1990.
15. Edward G. Coffman Jr., János Csirik, Lajos Rónyai, and Ambrus Zsbán. Random-order bin packing. *Discrete Applied Mathematics*, 156(14):2810–1816, 2008.
16. Bala Kalyanasundaram and Kirk Pruhs. *Online Algorithms: The State of the Art, eds. A. Fiat and G. Woeginger*, chapter 12: Online network optimization problems. Lecture Notes in Computer Science 1442, 1998.
17. Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.
18. Claire Kenyon. Best-fit bin-packing with random order. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 359–364, 1996.
19. Katalin Kolossa. On the on-line chromatic number of the family of on-line 3-chromatic graphs. *Discrete Mathematics*, 150:205–230, 1996.
20. Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.